

## ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №2 ЗАПИСИ С ВАРИАНТАМИ. ОБРАБОТКА ТАБЛИЦ

### 1) описание условия задачи;

Создать таблицу, содержащую не менее 40 записей с вариантной частью. Произвести поиск информации по вариантному полю. Упорядочить таблицу, по возрастанию ключей (где ключ – любое невариантное поле по выбору программиста), используя: а) исходную таблицу; б) массив ключей, используя 2 разных алгоритма сортировки (простой, ускоренный). Оценить эффективность этих алгоритмов (по времени и по используемому объему памяти) при различной реализации программы, то есть, в случаях а) и б). Обосновать выбор алгоритмов сортировки.

Вариант-1: Ввести список литературы, содержащий фамилию автора, название книги, издательство, количество страниц, вид литературы (1: техническая – отрасль, отечественная, переводная, год издания; 2: художественная – роман, пьеса, поэзия; 3: детская – минимальный возраст, сказки, стихи). Вывести список всех романов указанного автора.

### 2) техническое задание

- описание исходных данных и результатов (то есть, типы, форматы, точность, способ передачи, ограничения):

Исходные данные:

- файл, существует, разрешено чтение (также необходимо разрешение на запись в некоторых пунктах меню программы), содержит валидные записи в количестве от 40 до 100 штук
- выбор пункта меню:
  1. упорядочить данные в файле по возрастанию количества страниц
  2. добавить записи в конец списка
  3. удалить записи по количеству страниц
  4. найти все романы указанного пользователем автора
  5. вывод отсортированной таблицы ключей при несортированной исходной таблице
  6. вывод упорядоченной исходной таблицы
  7. вывод исходной таблицы в упорядоченном виде, используя упорядоченную таблицу ключей
  8. вывод исходной таблицы
  9. вывод таблицы ключей

10. вывод таблицы результатов сравнения эффективности работы программы при обработке данных в исходной таблице и в таблице ключей при использовании различных алгоритмов сортировок

11. выход из программы

➤ состав валидной записи:

- ✚ фамилия автора - от 1 до 25-и символов, не содержащих перенос строки
- ✚ название книги - от 1 до 25-и символов, не содержащих перенос строки
- ✚ издательство - от 1 до 25-и символов, не содержащих перенос строки
- ✚ количество страниц - целое число от 1 до 2 000 000 000
- ✚ вид литературы (ВЛ) - 1 – техническая, 2 – художественная или 3 – детская
- ✚ отрасль (если ВЛ = 1) - от 1 до 25-и символов, не содержащих перенос строки
- ✚ «отечественная» / «переводная» (если ВЛ = 1)
- ✚ год издания (если ВЛ = 1) - целое число от 0 до 1 000 000
- ✚ «роман» / «пьеса» / «поэзия» (если ВЛ = 2)
- ✚ минимальный возраст (если ВЛ = 3) - целое число от 0 до 17
- ✚ «сказки» / «стихи» (если ВЛ = 3)

Выходные данные:

- ✓ таблица ключей (в зависимости от пункта меню может быть отсортированная или неотсортированная)
- ✓ полная таблица (в зависимости от пункта меню может быть отсортированная, неотсортированная или выведенная отсортированной благодаря отсортированной таблице ключей)
- ✓ результаты сравнения сортировок разных таблиц двумя способами и объёма занимаемой памяти
- ✓ добавление записи в файл
- ✓ отсортированный файл
- ✓ удаление записи из файла
- ✓ найденные романы по автору
- описание задачи, реализуемой программой:

Задача, реализуемая программой, заключается в создании и управлении таблицей литературных произведений, содержащей информацию о книгах, включая фамилию автора, название книги, издательство, количество страниц и вид литературы. Программа предоставляет следующие функции:

1. Упорядочивание данных: Пользователь может выбрать упорядочивание данных в таблице по возрастанию количества страниц в книгах. Это позволяет легко находить самые объемные или самые краткие произведения.

2. Добавление записей: Пользователь может добавлять новые записи о книгах в конец списка. Это полезно для постоянного обновления таблицы с новыми произведениями.

3. Удаление записей: Пользователь может удалять записи о книгах на основе количества страниц. Например, можно удалить все книги с определенным количеством страниц.

4. Поиск романов автора: Пользователь может найти все романы определенного автора, что позволяет легко определить литературное наследие автора в жанре романа.

5. Сортировка таблицы ключей: Программа выводит упорядоченную таблицу ключей (например, номера страниц) при несортированной исходной таблице. Это может помочь в оптимизации поиска.

6. Вывод упорядоченной исходной таблицы: Пользователь может получить упорядоченную исходную таблицу на основе выбранного поля (например, количество страниц).

7. Вывод исходной таблицы с использованием таблицы ключей: Программа позволяет вывести исходную таблицу в упорядоченном виде, используя упорядоченную таблицу ключей. Это может быть полезно для оптимизации работы с данными.

8. Вывод исходной таблицы и таблицы ключей: Пользователь может просматривать исходную таблицу и таблицу ключей, чтобы легко сравнивать данные.

9. Вывод таблицы результатов сравнения эффективности алгоритмов сортировки: Программа сравнивает эффективность разных алгоритмов сортировки при обработке данных в исходной таблице и в таблице ключей. Это позволяет выбрать наиболее эффективный алгоритм для данной задачи.

10. Выход из программы: Пользователь может завершить выполнение программы.

Выбор алгоритмов сортировки зависит от объема данных и требований к производительности. Простой алгоритм сортировки (например, сортировка выбором) может использоваться для небольших объемов данных, в то время как более быстрые алгоритмы (например, быстрая сортировка) могут быть предпочтительными для больших объемов данных. Программа оценивает эффективность каждого алгоритма по времени выполнения и используемой памяти и предоставляет результаты для выбора наиболее подходящего алгоритма.

- способ обращения к программе:

Для запуска программы необходимо в терминале перейти в папку, содержащую исполняемый файл программы, и запустить его командой ./app.exe. Дальнейшее взаимодействие с программой происходит также в терминале.

- описание возможных аварийных ситуаций и ошибок пользователя:

<i>Ошибка пользователя</i>	<i>Ответ программы</i>
меню	
Некорректный ввод: На запрос меню ввести целое число от 1 до 11 (какое действие выбрать) пользователь ввел буквы.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Пожалуйста, попробуйте ввести снова.
Некорректный ввод: На запрос меню ввести целое число от 1 до 11 (какое действие выбрать) пользователь ввел число отличное от чисел от 1 до 11.	
Пустой ввод: На запрос меню ввести целое число от 1 до 11 (какое действие выбрать) пользователь нажал Enter.	
файл	
Пустой ввод: На запрос программы ввести название файла пользователь нажал Enter.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: пустой путь к файлу. Попробуйте ввести ещё раз.
Некорректный ввод:	Вывод сообщения об ошибке и

На запрос программы ввести название файла пользователь ввёл строку длиннее 25 символов.	повторный запрос ввода: Ошибка: слишком длинный путь к файлу. Попробуйте ввести ещё раз.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл файл, который не удалось открыть на чтение (файл закрыт для чтения).	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: не удалось открыть файл на чтение. Попробуйте ввести ещё раз.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл файл, который удалось открыть на чтение. Но затем пользователь выбрал один из пунктов меню, где программа должна изменить файл (1-3), а открыть файл на запись не удалось.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка открытия файла на запись. Пожалуйста, попробуйте ввести путь к файлу снова.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл название несуществующего файла.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: не удалось открыть файл на чтение. Попробуйте ввести ещё раз.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл название файла, содержащего некорректные записи.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка чтения списка из файла. Попробуйте ввести путь к файлу снова.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл название файла, содержащего слишком мало записей (меньше 40-а).	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: в файле слишком мало записей «количество_прочитанных_записей» (должно быть минимум 40). Попробуйте ввести путь к файлу снова.
Некорректный ввод: На запрос программы ввести название файла пользователь ввёл название файла, содержащего слишком много записей (больше 100-а).	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: в файле слишком много записей «количество_прочитанных_записей» (должно быть максимум 100). Попробуйте ввести путь к файлу снова.
Некорректный ввод: На запрос программы ввести	Вывод сообщения об ошибке и повторный запрос ввода:

название файла пользователь ввёл строку, которую не удалось прочитать.	Ошибка: не удалось прочитать путь к файлу. Попробуйте ввести ещё раз.
структура	
<p>Пустой ввод: На запрос программы ввести поле структуры пользователь нажал Enter.</p> <p>Некорректный ввод: На запрос программы пользователь ввёл слишком длинную строку.</p> <p>Некорректный ввод: На запрос программы ввести одну из строчек на выбор пользователь ввёл что-то другое.</p>	Вывод сообщения об ошибке и повторный запрос ввода.
<p>Некорректный ввод: На запрос программы ввести число пользователь ввёл буквы.</p> <p>Некорректный ввод: На запрос программы ввести число из диапазона пользователь ввёл число, не попадающее в диапазон.</p> <p>Некорректный ввод: На запрос программы ввести целое число пользователь ввёл действительное.</p>	
<p>Ошибка диапазона количества структур: Пользователь попытался удалить запись, когда их в файле осталось минимальное обрабатываемое количество (40).</p> <p>Ошибка диапазона количества структур: Пользователь попытался добавить запись, когда их в файле уже максимальное обрабатываемое количество (100).</p> <p>Некорректный ввод: Пользователь попытался удалить несуществующую запись.</p> <p>Некорректный ввод: Пользователь попытался найти несуществующую запись.</p>	
	Программа выводит сообщение об ошибке и выводит меню.

## 2) описание внутренних структур данных

```
// таблица

ft_table_t *table = ft_create_table();

//что выбрал пользователь в меню (целое число)

int user = 0;

//код возврата (целое число)

int rc = OK;

// переменная хранит имя файла (массив символов / строка)

wchar_t strk[MAX_LEN + 1];

// количество записей (целое число)

int n;

// массив, содержащий исходную таблицу (структура)

struct Book books[MAX_COUNT];

// массив, содержащий таблицу ключей (структура)

struct Book_key book_keys[MAX_COUNT];


//коды возврата (целые числа)

#define OK 0

#define ERROR 1

#define MY_ERROR 2


//максимальная длина строки (целое число)

#define MAX_LEN 25

//максимальное количество структур (целое число)

#define MAX_COUNT 100
```

```
//минимальное количество структур (целое число)

#define MIN_COUNT 40

//максимальное количество страниц (целое число)

#define MAX_LISTS 2000000000

//максимальное количество страниц (целое число)

#define MAX_YEAR 1000000

//максимальный минимальный возраст детской литературы (целое число)

#define MAX_CHILD_AGE 17

//количество сортировок (целое число)

#define COUNT_SORT 50

//размер очищаемого буфера (целое число)

#define SIZE_OF_BUF 100


//структура полной записи

struct Book

{

    wchar_t surname[MAX_LEN + 1];

    wchar_t name_book[MAX_LEN + 1];

    wchar_t mader[MAX_LEN + 1];

    int lists;

    int variant;//1, 2 or 3

    union spec // объединение разных видов литературы

    {

        struct texnical // структура технической литературы

        {
```



```

wchar_t otrasl[MAX_LEN + 1];

wchar_t where[MAX_LEN + 1];// отечественная/переводная

int year;

} texnic;

wchar_t draw[MAX_LEN + 1]; // роман/пьеса/поэзия

struct children // структура детской литературы
{
    int min_year;

    wchar_t what[MAX_LEN + 1];// сказки/стихи
} child;

} unic;

};

//структура - ключ

struct Book_key
{
    int start_ind;

    int lists;
};

```

### 3) описанный алгоритм:

При запуске программы выводится информация о ней и запрашивается название рабочего файла. После этого программа пытается считать файл в массив структур по шаблону таблицы и при ошибке просит заново ввести рабочий файл. После успешного прочтения рабочего файла выводится меню и запрашивается номер выбранного пункта, пока не будет введено валидное число.

Если пользователь ввёл число 1, вызывается функция сортировки данных в файле. В функции проверяется, что программа может писать в

файл, иначе запрашивается имя нового рабочего файла, в который программа сможет писать. Затем сортировкой выбором упорядочиваются элементы в полной таблице и таблице ключей, и с помощью отсортированной полной таблицы файл перезаписывается структурами в правильном порядке. Файл закрывается, и управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 2, вызывается функция добавления записи в конец файла. Программа проверяет не находится ли уже в файле максимальное количество записей (100) и, если это число достигнуто, выводит ошибку. Иначе программа пытается открыть файл на дозапись и, если это не получается, запрашивает имя нового файла, который сможет открыть в этом режиме. Затем у пользователя запрашивается добавляемая запись. Все поля тщательно проверяются, и при ошибке программа просит заново ввести неверное поле. После успешного ввода пользователем новой записи программа демонстрирует её пользователю, дозаписывает в файл и возвращает управление в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 3, вызывается функция удаления первой записи в файле по полю «количество страниц». Программа проверяет не находится ли уже в файле минимальное количество записей (40) и, если это число есть, выводит ошибку. Иначе программа пытается открыть файл на перезапись и, если это не получается, запрашивает имя нового файла, который сможет открыть в этом режиме. Затем у пользователя запрашивается значение поля (количества страниц больше нуля и меньше и равно 2 млн.) записи, которую нужно удалить. Если таких элементов несколько, удаляется первый. Затем программа проходит по всем записям и находит первую подходящую, выводит её пользователю и сдвигает все записи после найденной на один к началу, количество записей уменьшается на 1. Если ни одна запись не подошла, выводится: «Запись с таким значением поля отсутствует». С помощью полной таблицы файл перезаписывается и закрывается. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 4, вызывается функция поиска всех романов указанного пользователем автора. Программа запрашивает автора, романы которого требуется найти и проверяет введённую строчку. Пока строка содержит ошибку (пустая / слишком длинная строка) вновь запрашивается фамилия автора. Затем программа проходит по всей полной

таблице и находит совпадения по автору и тому, что он написал роман и выводит найденное на экран в таблицу. Если ни одна запись не подошла, выводится, что «Романы этого автора не найдены». Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 5, вызывается функция вывода отсортированной таблицы ключей. Программа сортирует таблицу ключей выбором (после каждого прохода по массиву находится максимальный элемент и ставится в конец просматриваемой области, затем эта область сужается и всё повторяется, пока длина области не становится единичной) и выводит её пользователю в виде таблицы. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 6, вызывается функция вывода отсортированной полной таблицы. Программа сортирует полную таблицу выбором (после каждого прохода по массиву находится максимальный элемент и ставится в конец просматриваемой области, затем эта область сужается и всё повторяется, пока длина области не становится единичной), исправляет индексы в таблице ключей и выводит массив полных записей пользователю в виде таблицы. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 7, вызывается функция вывода упорядоченной исходной таблицы, используя упорядоченную таблицу ключей. Программа сортирует таблицу ключей выбором (после каждого прохода по массиву находится максимальный элемент и ставится в конец просматриваемой области, затем эта область сужается и всё повторяется, пока длина области не становится единичной) и выводит пользователю массив полных записей по индексам из отсортированной таблицы ключей в виде таблицы. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 8, вызывается функция вывода полной таблицы. Программа выводит массив полных записей пользователю в виде таблицы. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 9, вызывается функция вывода таблицы ключей. Программа выводит массив записей-ключей пользователю в виде

таблицы. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 10, вызывается функция вывода таблицы сравнения разных сортировок. Подсчитывается время сортировки выбором полной таблицы: создаётся копия полной таблицы, сортируется методом выбора (после каждого прохода по массиву находится максимальный элемент и ставится в конец просматриваемой области, затем эта область сужается и всё повторяется, пока длина области не становится единичной) и вычисляется время сортировки (в тактах), это повторяется 50 раз, затем высчитывается среднее время. Подсчитывается время рекурсивной сортировки полной таблицы: создаётся копия полной таблицы, сортируется рекурсивным методом (массив разбивается на две части, пока элементы с обеих сторон меньше (слева) и больше (справа) срединного элемента границы сдвигаются, оставшиеся в сдвинувшейся области элементы меняются местами, потом эта же функция рекурсивной сортировки вызывается для правой, пока нижняя граница области меньше индекса конца всего массива, и левой частей, пока верхняя граница области больше индекса начала всего массива) и вычисляется время сортировки (в тактах), это повторяется 50 раз, затем высчитывается среднее время. Подсчитывается время сортировки выбором таблицы ключей: создаётся копия таблицы ключей, сортируется методом выбора (после каждого прохода по массиву находится максимальный элемент и ставится в конец просматриваемой области, затем эта область сужается и всё повторяется, пока длина области не становится единичной) и вычисляется время сортировки (в тактах), это повторяется 50 раз, затем высчитывается среднее время. Подсчитывается время рекурсивной сортировки таблицы ключей: создаётся копия таблицы ключей, сортируется рекурсивным методом (массив разбивается на две части, пока элементы с обеих сторон меньше (слева) и больше (справа) срединного элемента границы сдвигаются, оставшиеся в сдвинувшейся области элементы меняются местами, потом эта же функция рекурсивной сортировки вызывается для правой, пока нижняя граница области меньше индекса конца всего массива, и левой частей, пока верхняя граница области больше индекса начала всего массива) и вычисляется время сортировки (в тактах), это повторяется 50 раз, затем высчитывается среднее время. После этого результаты замеров времени (в тактах процессора), размеры занимаемой памяти, вычисленной с помощью функции `sizeof` в байтах (массива полных записей для исходной таблицы и массива полных записей + массива записей-ключей для таблицы ключей) и сравнение сортировок по

времени и по памяти таблиц полной и ключей (в процентах) выводятся виде таблицы на экран. Затем управление возвращается в главную функцию, где вновь выводится меню и запрашивается выбранный пункт.

Если пользователь ввёл число 11, программа завершается.

5) набор тестов, с указанием, что проверяется:

Тестовые данные	Что проверяется	Вывод программы
file.txt 1 (сортировка данных в файле) <ul style="list-style-type: none"> <li>• 40 элементов уже отсортированы</li> <li>• 40 элементов рандомно расположены</li> <li>• 40 элементов отсортированы в обратном порядке</li> <li>• 70 элементов уже отсортированы</li> <li>• 70 элементов рандомно расположены</li> <li>• 70 элементов отсортированы в обратном порядке</li> <li>• 40 элементов уже отсортированы</li> <li>• 100 элементов рандомно расположены</li> <li>• 100 элементов отсортированы в обратном порядке</li> </ul>	Проверка правильности сортировки данных в файле (минимальное, среднее и максимальное количество разной начальной отсортированности)	Данные в файле отсортированы
file.txt 2 (добавление структуры)	Проверка правильности добавления структуры.	Структура добавлена в конец файла.
file.txt 3 (удаление структуры по полю) <ul style="list-style-type: none"> <li>• подходящая структура 1</li> <li>• подходящих структур несколько</li> </ul>	Проверка правильности удаления структуры.	Первая подходящая структура удалена из файла.

file.txt 4 (поиск романов автора) <ul style="list-style-type: none"> <li>• подходящая структура 1</li> <li>• подходящих структур несколько</li> </ul>	Проверка правильности поиска структур.	Все подходящие структуры выведены на экран.
file.txt 5 (вывод отсортированной таблицы ключей) <ul style="list-style-type: none"> <li>• 40 элементов уже отсортированы</li> <li>• 40 элементов рандомно расположены</li> <li>• 40 элементов отсортированы в обратном порядке</li> <li>• 70 элементов уже отсортированы</li> <li>• 70 элементов рандомно расположены</li> <li>• 70 элементов отсортированы в обратном порядке</li> <li>• 40 элементов уже отсортированы</li> <li>• 100 элементов рандомно расположены</li> <li>• 100 элементов отсортированы в обратном порядке</li> </ul>	Проверка правильности сортировки таблицы ключей	Отсортированная таблица ключей выведена на экран.
file.txt 6 (вывод упорядоченной исходной таблицы) <ul style="list-style-type: none"> <li>• 40 элементов уже отсортированы</li> <li>• 40 элементов рандомно расположены</li> <li>• 40 элементов отсортированы в обратном порядке</li> <li>• 70 элементов уже</li> </ul>	Проверка правильности сортировки исходной таблицы	Отсортированная исходная таблица выведена на экран.

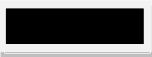
<p>отсортированы</p> <ul style="list-style-type: none"> <li>• 70 элементов рандомно расположены</li> <li>• 70 элементов отсортированы в обратном порядке</li> <li>• 40 элементов уже отсортированы</li> <li>• 100 элементов рандомно расположены</li> <li>• 100 элементов отсортированы в обратном порядке</li> </ul>		
<p>file.txt 7 (вывод упорядоченной исходной таблицы по таблице ключей)</p> <ul style="list-style-type: none"> <li>• 40 элементов уже отсортированы</li> <li>• 40 элементов рандомно расположены</li> <li>• 40 элементов отсортированы в обратном порядке</li> <li>• 70 элементов уже отсортированы</li> <li>• 70 элементов рандомно расположены</li> <li>• 70 элементов отсортированы в обратном порядке</li> <li>• 40 элементов уже отсортированы</li> <li>• 100 элементов рандомно расположены</li> <li>• 100 элементов отсортированы в обратном порядке</li> </ul>	<p>Проверка правильности вывода отсортированной исходной таблицы по таблице ключей</p>	<p>Отсортированная исходная таблица выведена на экран по таблице ключей.</p>
<p>file.txt 8 (вывод исходной таблицы)</p>	<p>Проверка правильности</p>	<p>Выведена исходная таблица.</p>

<ul style="list-style-type: none"> <li>• 40 элементов</li> <li>• 70 элементов</li> <li>• 100 элементов</li> </ul>	вывода исходной таблицы.	
file.txt 9 (вывод таблицы ключей) <ul style="list-style-type: none"> <li>• 40 элементов</li> <li>• 70 элементов</li> <li>• 100 элементов</li> </ul>	Проверка правильности вывода таблицы ключей.	Выведена таблица ключей.
file.txt 10 (вывод таблицы сравнения разных сортировок и занимаемой памяти) <ul style="list-style-type: none"> <li>• 40 элементов уже отсортированы</li> <li>• 40 элементов рандомно расположены</li> <li>• 40 элементов отсортированы в обратном порядке</li> <li>• 70 элементов уже отсортированы</li> <li>• 70 элементов рандомно расположены</li> <li>• 70 элементов отсортированы в обратном порядке</li> <li>• 40 элементов уже отсортированы</li> <li>• 100 элементов рандомно расположены</li> <li>• 100 элементов отсортированы в обратном порядке</li> </ul>	Проверка правильности вывода таблицы сравнения разных сортировок и занимаемой памяти.	Выведена таблица сравнения разных сортировок и занимаемая память.
file.txt 11	Проверка правильности завершения работы.	Программа завершила работу.
Ошибки меню		
file.txt авс	На запрос меню ввести целое число от 1 до 11 (какое действие	Вывод сообщения об ошибке и повторный запрос ввода:



	выбрать) пользователь ввел буквы.	Ошибка ввода. Пожалуйста, попробуйте ввести снова.
file.txt 12	На запрос меню ввести целое число от 1 до 11 (какое действие выбрать) пользователь ввел число отличное от чисел от 1 до 11.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Пожалуйста, попробуйте ввести снова.
file.txt	Пустой ввод: На запрос меню ввести целое число от 1 до 11 (какое действие выбрать) пользователь нажал Enter.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Пожалуйста, попробуйте ввести снова.
файл		
	Пустой ввод: На запрос программы ввести название файла пользователь нажал Enter.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: пустой путь к файлу. Попробуйте ввести ещё раз.
Ауграшгукршгаркугшпрк23рпгш цркагуш	На запрос программы ввести название файла пользователь ввёл строку длиннее 25 символов.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: слишком длинный путь к файлу. Попробуйте ввести ещё раз.
File2.txt	На запрос программы ввести название файла пользователь ввёл файл, который не удалось открыть на чтение (файл	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: не удалось открыть файл на чтение. Попробуйте

	закрыт для чтения).	ввести ещё раз.
File3.txt 1	На запрос программы ввести название файла пользователь ввёл файл, который удалось открыть на чтение. Но затем пользователь выбрал один из пунктов меню, где программа должна изменить файл (1-3), а открыть файл на запись не удалось.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка открытия файла на запись. Пожалуйста, попробуйте ввести путь к файлу снова.
File4.txt	На запрос программы ввести название файла пользователь ввёл название несуществующего файла.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: не удалось открыть файл на чтение. Попробуйте ввести ещё раз.
File5.txt	На запрос программы ввести название файла пользователь ввёл название файла, содержащего некорректные записи.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка чтения списка из файла. Попробуйте ввести путь к файлу снова.
File6.txt	На запрос программы ввести название файла пользователь ввёл название файла, содержащего слишком мало записей (меньше 40-а).	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: в файле слишком мало записей «количество_прочитанных_записей» (должно быть минимум 40).

		Попробуйте ввести путь к файлу снова.
File7.txt	На запрос программы ввести название файла пользователь ввёл название файла, содержащего слишком много записей (больше 100-а).	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: в файле слишком много записей «количество_прочитанных_записей» (должно быть максимум 100). Попробуйте ввести путь к файлу снова.
	На запрос программы ввести название файла пользователь ввёл строчку, которую не удалось прочитать.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка: не удалось прочитать путь к файлу. Попробуйте ввести ещё раз.
структура		
	На запрос программы ввести поле структуры пользователь нажал Enter.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Попробуйте ввести снова:
Uweifhewiufhweiufhiewuhfuiewfh wefheufiheiufhiewfhiewuhfiuewhf iuehwfiuhwefu (максимум 25)	На запрос программы пользователь ввёл слишком длинную строчку.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Попробуйте ввести снова:
Страна (требовалось ввести «сказки» или «стихи»)	На запрос программы ввести одну из строчек на выбор пользователь ввёл что-то другое.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода. Попробуйте ввести снова

		('сказки'/'стихи'):
авс	На запрос программы ввести число пользователь ввёл буквы.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода вида литературы. Попробуйте ввести снова (1/2/3):
4 (Требовалось число от 1 до 3)	На запрос программы ввести число из диапазона пользователь ввёл число, не попадающее в диапазон.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода вида литературы. Попробуйте ввести снова (1/2/3):
3.14	На запрос программы ввести целое число пользователь ввёл действительное.	Вывод сообщения об ошибке и повторный запрос ввода: Ошибка ввода вида литературы. Попробуйте ввести снова (1/2/3):
File10.txt 3	Пользователь попытался удалить запись, когда их в файле осталось минимальное обрабатываемое количество (40).	Вывод сообщения об ошибке и повторный запрос ввода: Невозможно удалить поле. Программа работает с таблицами больше или равно 40 записей
File11.txt 2	Пользователь попытался добавить запись, когда их в файле уже максимальное обрабатываемое количество (100).	Вывод сообщения об ошибке и повторный запрос ввода: В файле уже максимальное количество записей (100). Добавление невозможно.

File12.txt 3 234 (записи с таким полем не существует)	Пользователь попытался удалить несуществующую запись.	Вывод сообщения об ошибке и вывод меню: Запись с таким значением поля отсутствует.
File13.txt 4 William (романов с таким автором не существует)	Пользователь попытался найти несуществующую запись.	Вывод сообщения об ошибке и вывод меню: Романы этого автора не найдены.

б) оценка эффективности:

\* Время выводится в тактах процессора, память в байтах, количество замеров времени = 50 раз.

✓ 40 элементов (минимальное количество) уже отсортированы:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	214,680000	162,300000	21280,000000
Таблица ключей	17,320000	6,820000	21600,000000
Выигрыш от таблицы ключей в процентах	1139,49%	2279,77%	-1,48%

✓ 40 элементов (минимальное количество) расположены рандомно:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	235,400000	352,040000	21280,000000
Таблица ключей	18,480000	10,700000	21600,000000
Выигрыш от таблицы ключей в процентах	1173,80%	3190,09%	-1,48%

✓ 40 элементов (минимальное количество) уже отсортированы в обратном порядке:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	218,820000	251,920000	21280,000000

Таблица ключей	17,160000	9,720000	21600,000000
Выигрыш от таблицы ключей в процентах	1175,17%	2491,77%	-1,48%

✓ 70 элементов (среднее количество) уже отсортированы:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	437,580000	403,680000	37240,000000
Таблица ключей	38,020000	13,240000	37800,000000
Выигрыш от таблицы ключей в процентах	1050,92%	2948,94%	-1,48%

✓ 70 элементов (среднее количество) расположены рандомно:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	484,460000	788,580000	37240,000000
Таблица ключей	45,620000	20,400000	37800,000000
Выигрыш от таблицы ключей в процентах	961,95%	3765,59%	-1,48%

✓ 70 элементов (среднее количество) уже отсортированы в обратном порядке:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	395,800000	537,920000	37240,000000
Таблица ключей	46,040000	16,320000	37800,000000
Выигрыш от таблицы ключей в процентах	759,69%	3196,08%	-1,48%

✓ 100 элементов (максимальное количество) уже отсортированы:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	592,020000	622,300000	53200,000000
Таблица ключей	90,120000	20,200000	54000,000000
Выигрыш от таблицы ключей в процентах	556,92%	2980,69%	-1,48%

✓ 100 элементов (максимальное количество) расположены  
рандомно:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	912,940000	1510,040000	53200,000000
Таблица ключей	127,840000	200,640000	54000,000000
Выигрыш от таблицы ключей в процентах	614,13%	652,61%	-1,48%

✓ 100 элементов (максимальное количество) уже  
отсортированы в обратном порядке:

Таблица	Сорт. выбором	Рекурсивная сорт.	Объем памяти
Полная таблица	633,820000	929,500000	53200,000000
Таблица ключей	81,380000	23,360000	54000,000000
Выигрыш от таблицы ключей в процентах	678,84%	3879,02%	-1,48%

7) выводы по проделанной работе:

Лабораторная работа №2 «Записи с вариантами. Обработка таблиц» помогла осознать, что для хранения записей с большим количеством полей можно использовать специальные типы данных (например, структуры в Си), хранящие информацию об одном объекте как единое целое. А для хранения информации разных типов в одном месте памяти также удобно использовать специальные типы данных (например, объединения в Си). Эти типы можно комбинировать и вкладывать друг в друга в зависимости от задачи.

В данной задаче программа, использующая таблицу ключей для сортировки, занимает на 1,48% памяти больше, но при этом работает во много раз быстрее.

8) ответы на вопросы:

1.Как выделяется память под вариантную часть записи?

При хранении вариантной части записи разные варианты как бы «накладываются» друг на друга, они разделяют одну область памяти и в зависимости от значения специального целочисленного поля, указывающего

тип хранимых в настоящий момент данных, область содержит тот или иной вариант записи. При этом размер выделенной памяти под вариантную часть равен максимальному возможному варианту её заполнения.

2. Что будет, если в вариантную часть ввести данные, несоответствующие описанным?

Результат такого действия будет зависеть от множества параметров и в общем случае определить что случится невозможно. Ввод в вариантную часть данных, несоответствующих описанным, приведёт к неопределённому поведению программы.

3. Кто должен следить за правильностью выполнения операций с вариантной частью записи?

За правильностью выполнения операций с вариантной частью записи должен следить программист.

4. Что представляет собой таблица ключей, зачем она нужна?

Таблица ключей – это массив структур, содержащий два поля: индекс исходной таблицы и ключевое поле. Она может помочь в оптимизации сортировки / поиска, что экономит время, однако при этом количество памяти, занимаемой программой, увеличивается.

5. В каких случаях эффективнее обрабатывать данные в самой таблице, а когда – использовать таблицу ключей?

Эффективнее обрабатывать данные в исходной таблице, когда она одна её запись занимает небольшой объём памяти, при этом, если разница в размере занимаемой памяти между самой таблицей и таблицей ключей велика, эффективнее использовать таблицу ключей, так как придётся оперировать меньшим объёмом памяти, что экономит время.

Также необходимо ориентироваться на задачу: что необходимо экономить, а чем можно пренебречь – временем или памятью? Если важен размер памяти, то таблицу ключей лучше не использовать, а, если важно время работы, таблица ключей станет хорошим решением.

6. Какие способы сортировки предпочтительнее для обработки таблиц и почему?

1. Сортировка слиянием (Merge Sort): Этот метод обычно предпочтителен, потому что он стабильный и эффективен даже для больших



массивов данных. Стабильность важна, когда вы хотите сохранить относительный порядок элементов с одинаковыми ключами. Он также гарантирует временную сложность  $O(n \cdot \log(n))$  в худшем и среднем случае.

2. Быстрая сортировка (Quick Sort): Quick Sort также может быть эффективным методом для сортировки массивов структур. Он хорошо справляется с сортировкой в среднем случае и имеет среднюю временную сложность  $O(n \cdot \log(n))$ . Однако, в худшем случае, когда разделение происходит не сбалансированно, временная сложность может стать  $O(n^2)$ .

3. Сортировка вставками (Insertion Sort): Если ваш массив структур небольшой или уже частично отсортирован, сортировка вставками может быть эффективным выбором. Она имеет простой код и может быть эффективной для небольших массивов данных.

4. Сортировка пузырьком (Bubble Sort): Как и сортировка вставками, сортировка пузырьком может быть подходящей для небольших массивов. Однако она менее эффективна по сравнению с более эффективными алгоритмами, такими как Quick Sort и Merge Sort.

Выбор конкретного метода сортировки зависит от вашей конкретной задачи и характеристик данных. Если вы сортируете большие массивы данных и вам важна стабильность, Merge Sort часто является хорошим выбором. Если вы ожидаете небольшие массивы или знаете, что данные частично упорядочены, можно рассмотреть более простые алгоритмы, такие как Insertion Sort или Bubble Sort.